International Journal of Production Research

ISSN: 0020-7543 (Print) 1366-588X (Online) Journal homepage: http://www.tandfonline.com/loi/tprs20

Reactive scheduling in a dynamic and stochastic FMS environment

Ihsan Sabuncuoglu & Omer Batuhan Kizilisik

To cite this article: Ihsan Sabuncuoglu & Omer Batuhan Kizilisik (2003) Reactive scheduling in a dynamic and stochastic FMS environment, International Journal of Production Research, 41:17, 4211-4231, DOI: 10.1080/0020754031000149202

To link to this article: http://dx.doi.org/10.1080/0020754031000149202

Published online: 14 Nov 2010.

INT. J. PROD. RES., 2003, VOL. 41, NO. 17, 4211–4231

Taylor & Francis
Taylor & Francis Group

Reactive scheduling in a dynamic and stochastic FMS environment

IHSAN SABUNCUOGLU†* and OMER BATUHAN KIZILISIK†

In this paper, we study the reactive scheduling problems in a dynamic and stochastic manufacturing environment. Specifically, we develop a simulation-based scheduling system for flexible manufacturing systems. We also propose several reactive scheduling policies (i.e. when-to-schedule and how-to-schedule policies) and test their performances under various experimental conditions, processing time variations, and machine breakdowns. Moreover, we compare offline and online scheduling schemes in a dynamic manufacturing environment. The results of extensive simulation experiments indicate that the variable-time-response is better than the fixed-time-response. The full scheduling scheme generally performs better than the partial scheduling. Finally, the online scheduling is more robust to uncertainty and variations in processing times than the optimum-seeking offline scheduling. A comprehensive bibliography is also provided in the paper.

1. Introduction

Scheduling is a decision making process that concerns the allocation of limited resources (machines, material handling equipment, operators, tools, fixtures, etc) to competing tasks (operations of jobs) over time with the goal of optimizing one or more objectives. The output of this decision process is time/machine/operation assignments.

In classical scheduling theory, the problems are formulated as the mathematical model and are solved either by exact or heuristic procedures. Since most of the scheduling problems are NP-hard (i.e. mathematically intractable), researchers can only handle the deterministic and static versions of these problems. In practice, however, real systems are stochastic and dynamic (e.g. dynamic job arrivals and unexpected disruptions in the form of breakdowns, order cancellations, etc) such that these optimum schedules can easily become infeasible even shortly after they are released to the shop floor. For that reason, maintaining a feasible schedule alone can sometimes be the only goal of scheduling in practice.

In general, there are two main elements in any scheduling system: *schedule generation* and *control*. Schedule generation is viewed as the planning module (or *predictive mechanism*) that determines the planned start and completion times of operations of the jobs. In contrast, the control element (*reactive mechanism*) concerns updating schedules or reacting to unexpected random events. In other words, this element monitors the execution of the schedule and revises it to cope with unexpected events such as machine breakdowns, arrival of hot jobs, etc. In practice,

Revision received 1 March 2003.
†Industrial Engineering Department, Bilkent University, 06533 Ankara, Turkey.
*To whom correspondence should be addressed. e-mail: sabun@bilkent.edu.tr

International Journal of Production Research ISSN 0020–7543 print/ISSN 1366–588X online
http://www.tandf.co.uk/journals
DOI: 10.1080/0020754031000149202

the performance of predetermined schedules degrades so quickly that an appropriate reactive action should be taken to return the system back to the planned or desired performance. Although the control (or reactive) aspect is the very important, it has not been adequately studied in the literature.

In this paper, we study the reactive scheduling problems in a dynamic and stochastic FMS (Flexible Manufacturing System). We consider a flexible manufacturing system because it provides a multi-machine environment with the capability of performing several different types of operations, and its material handling system. Typically, an FMS consists of a group of numerically controlled (NC) machines connected by an automated guided vehicle (AGV) system. The same system was also used in the previous studies (Sabuncuoglu and Karabuk 1998, 1999). In fact, this paper is a sequel to these previous studies. In Sabuncuoglu and Karabuk (1998), we developed a schedule generation mechanism using the filtered beam search and compared its performance with other scheduling methods. In Sabuncuoglu and Karabuk (1999), we analyzed the reactive scheduling problems in a static environment. In this study, we develop the simulation module and test the performance of several reactive policies (*when-to-schedule* and *how-to-schedule* reactive policies) under various experimental conditions. We also compare the online and offline scheduling approaches. Here, *offline scheduling* refers to all available jobs being scheduled all at once for the entire planning horizon. In *online scheduling*, one schedule is made at a time when it is needed according to the change in the system conditions (i.e. it is constructed over time, not all schedules at once). Priority dispatching is a good example of online scheduling.

The rest of the paper is organized as follows. In section 2, we summarize the relevant literature on reactive scheduling. In section 3, we describe the proposed scheduling system. In section 4, we present the proposed reactive scheduling policies. This is followed by the system considerations and experimental design in section 5. The results of simulation experiments are presented in section 6. The paper ends with concluding remarks and suggestions for further research in section 7.

2. Literature review

In the literature, there are a number of studies that analyze scheduling problems in a dynamic and stochastic environment and propose reactive policies for the effective shop floor control systems. This research work can be classified into: (1) an analytical approach, (2) simulation-based experimental studies and iterative simulation approaches, and (3) artificial intelligence and knowledge based systems.

Early research work on reactive scheduling used the analytical approach to address the dynamic nature of scheduling problems. In this approach, a series of *static and deterministic* problems are solved and their solutions are implemented in a rolling horizon basis. This is also called the *scheduling/rescheduling* approach. The first study in this area is probably that of Nelson and Holloway (1977), who propose a scheduling system for a job shop with intermittent job arrivals. Their scheduling algorithm is based on the multi-pass heuristic that generates schedules at each time period. This study is followed by the studies of Muhlemann *et al.* (1982) and Yamamoto and Nof (1985) who investigated rescheduling policies under random machine breakdowns. Later, Bean *et al.* (1991) proposed a match-up approach that reconstructs the schedule to match up with the preschedule at some future time. The proposed method was found superior to the preplanned static schedule and dispatching rules under various experimental conditions. In another study, Wu *et al.* (1993)

developed a local search algorithm to reschedule all unprocessed jobs right after a machine breakdown occurs. This method can also identify a match-up schedule if one exists. Church and Uzsoy (1992) studied the problem of rescheduling in a single machine environment with dynamic job arrivals. The authors developed periodic and event driven scheduling policies. Similarly, Ovacik and Uzsoy (1994) proposed several rolling horizon procedures that perform better than myopic heuristics. In another study, Sabuncuoglu and Karabuk (1999) investigated the scheduling rescheduling problem in an FMS environment. The authors developed a filtered beam search algorithm and tested several reactive scheduling policies in response to machine breakdowns and processing time variations. Later, Sabuncuoglu and Bayiz (2000) examined the reactive scheduling problems under various shop floor configurations (e.g. system size and load allocations).

There are also studies in the literature that proposes robust scheduling methods (Leon *et al*. 1994; Daniels and Kouvelis 1995; Wu and Storer 1994). Here, the goal is to develop a schedule whose performance does not degrade much in the face of disruptions. Similarly, Mehta and Uzsoy (1998, 1999) propose the methods that can generate stable schedules. A schedule whose realization does not deviate from the original schedule is called stable. The common denominator in all these applications is to cope with unexpected disruptions in a more effective way. This is currently an important research direction being pursued by various researches.

A second area in which numerous publications have emerged in recent years is the application of artificial intelligence (AI) and knowledge based systems (KBS). The basic motivation of these applications is that each scheduling system is unique and therefore, a wide variety of technical expertise, system specific knowledge and human judgement must be considered for solving them. The work by Fox and Smith (1984), Smith *et al*. (1990), Shaw *et al*. (1992), and Dutta (1994) are good examples for this line of research. More detailed discussions of AI technology related to reactive scheduling can be found in Szelke and Kerr (1994). The related works by artificial neural networks are summarized in Sabuncuoglu (1998).

Since a typical scheduling environment is, in practice, dynamic and requires continuous updates, discrete event simulation models are also used for the reactive scheduling systems. For example, Kim and Kim (1994) propose a simulation based scheduling system with two major components: simulation mechanism and reactive control. The simulation mechanism evaluates various rules and selects the best one for a given job population and performance criterion. Later, Jeong and Kim (1998) refine the functions and modules of the previous scheduling mechanism and test several rescheduling policies. Kutanoglu and Sabuncuoglu (2001a) also propose a simulation-based scheduling system and test new reactive policies. Their computational experiments provide important information about the effectiveness of iterative simulation procedures. In the above studies, the scheduling mechanism is based on dispatching. In all these applications, the best scheduling rule is selected via simulation according to a selected criterion and it is implemented until the next scheduling point. The selection process is triggered either regularly at the beginning of discrete time periods or by random events (e.g. breakdowns etc). In our study, however, the emphasis is on generating a full schedule and finding ways of revising schedules in response to random disturbances. A simulation model is used to represent the dynamic and stochastic manufacturing environment. In practice, some manufacturing systems operate with dispatching policies whereas others employ a detailed schedule of the entire system. In our opinion, these two lines of research deal with the problem

using different but complementary approaches. Hence, there is a need for research in each direction. In another study, Kutanoglu and Sabuncuoglu (2001b) investigate the performance of four reactive policies under machine breakdowns. The results indicate that all re-routing is more robust against machine breakdowns. There are also studies that measure the effects of stochastic events and variations on scheduling rules (He *et al.* 1994; Lawrence and Sewell 1997; Svestka and Abumaziar 1997; Lejmi and Sabuncuoglu 2002). The results indicate that simple rules are quite robust under stochastic disturbances and variations in processing times and due dates.

In summary, there are a number of studies in the literature that investigate reactive scheduling problems and propose the methods to deal with contingencies (i.e. machine breakdowns). In most of these studies, a static environment (i.e. no new job arrivals) is assumed by the researchers who generate offline schedules. Those who consider a dynamic environment usually employ dispatching rules as a part of the online scheduling mechanism. It seems that offline scheduling algorithms perform better than online dispatching rules in static and deterministic environments. But, their relative performance is not generally known in real life conditions (i.e. stochastic and dynamic environments). The purpose of our study is to investigate this problem in a more general environment (i.e. an FMS) so that some earlier results can be verified, new findings can be added, and thus a more general picture can be drawn.

3. Proposed scheduling system

A successful implementation of a scheduling system in a dynamic environment usually requires either a real manufacturing facility or a simulation environment. In our case, we use the simulation approach since it is cost-effective way of creating dynamic conditions. The proposed scheduling system consists of three major components: scheduler or scheduling module, simulation model, and controller (figure 1). The scheduler is responsible for making all scheduling decisions. Given the system status and other relevant data (i.e. online, offline) it generates a partial or a complete schedule. The scheduling module generates schedules by considering machines, AGVs, finite buffer capacities, sequence, and routing flexibility, etc.

The simulation module uses two sets of input data: system-related data and values of environmental parameters. System-related data consist of the physical description of the manufacturing system (i.e. number of machines, AGVs). The arrival rate of jobs, parameters of stochastic events (i.e. machine breakdown rate), flexibility settings constitute the environmental parameters. In the simulation model, both the movement of AGVs and in-process storage capacity are represented in detail. The main task of the simulation model is to implement the scheduling decisions, which are generated by the scheduler. When an online scheduling policy is implemented, a resource triggers the controller upon completing a task, which invokes the scheduler. The scheduler makes a decision by applying scheduling rules and passes the decision to the controller. Then the controller sends this schedule to the simulation model for execution. The control module examines the state of the system at every discrete event and provides an appropriate course of action to be executed by the simulation model. The control module has the following tasks: keep up with the machine and AGV sequence in offline mode, resolve deadlock and implement scheduling policies. It is not easy to follow the exact start and completion times imposed by an offline algorithm in a dynamic and stochastic environment. However, machine processing sequences and AGV move sequences should follow

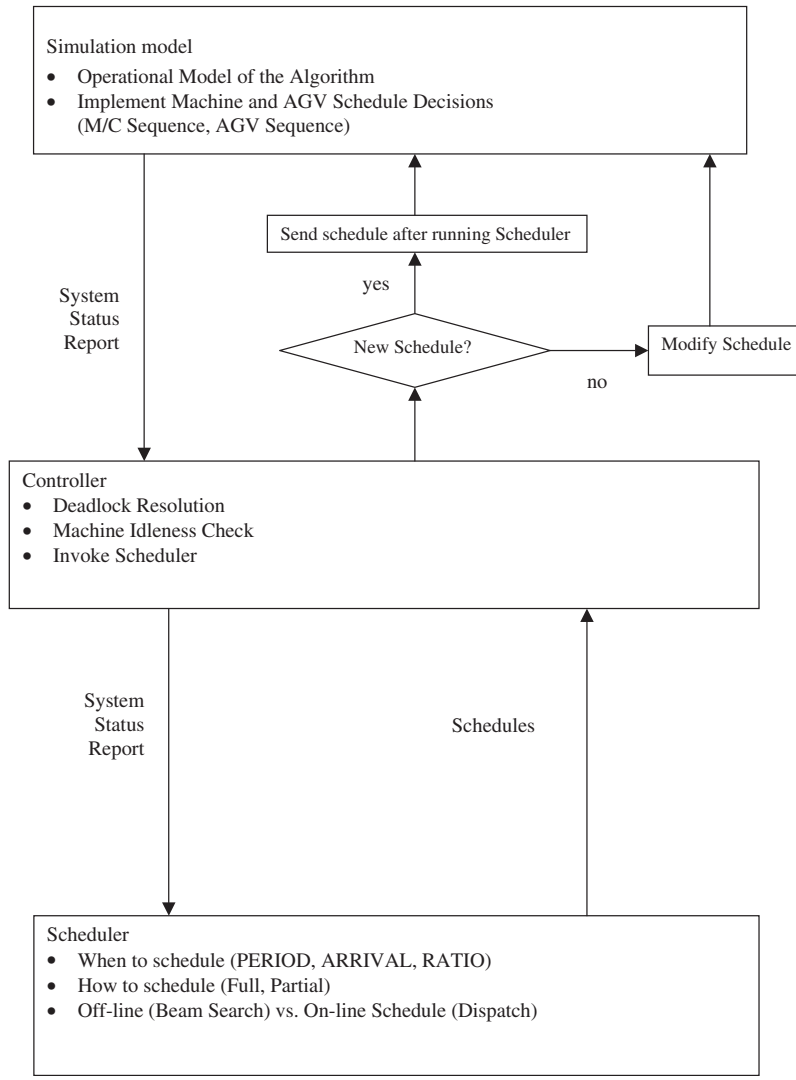


Figure 1. A simulation based scheduling system.

as close as possible the initial schedule. In most of the manufacturing systems, since in-storage capacity is limited, there is always a possibility for blocking (and locking) in the system due to finite capacities. This necessitates the use of effective control policies to avoid blocking in the system. As the third task, the controller is responsible for implementation of scheduling policies by considering the environmental conditions over time. In order to accomplish this task, the controller must either be supplied with the appropriate control policy or must simulate alternative policies and choose the best one. The first one is encountered in offline use of simulation, whereas the second stands for online use. The entire system is coded in C language and implemented in Unix environment.

The scheduling algorithm used in the schedule generation module is a heuristic based on the filtered beam search technique. Beam search is a fast and approximate

branch-and-bound (B&B) method that operates on a search tree. This partial enumeration technique uses heuristics to estimate a certain number of best paths, permanently pruning the rest. Since large parts of the search tree are pruned aggressively, solutions are obtained quickly. We used beam search not only because of its following properties. First, it is flexible enough to incorporate various system details (e.g. machines, AGVs, etc). Second, coding the algorithm is simple since it employs a breadth-first search strategy. Third, it is possible to generate schedules of various lengths since schedules are built progressively from the first operation to the last in a forward time frame. Hence, the proposed algorithm can generate partial schedules (for online applications) in addition to full schedules for the entire planning horizon.

In the algorithm, there are two major elements: (1) search tree representation to define a solution space and (2) application of a search methodology to find a good solution. The solution space is represented as a search tree where each node corresponds to a scheduling decision to be made and each unique path from the root node to any particular node defines a partial solution associated with that node. Leaf nodes specify complete solutions. The search tree is constructed such that various system resources (machines and AGVs), their capacities and flexibilities are taken into account at each layer.

After representing the solution space, an appropriate search procedure is used to find good solutions within this space. This search is performed in two stages. First, the stage involves elimination of unpromising nodes by a computational cheap method (i.e. local evaluation function). The remaining nodes (filterwidth) are then evaluated in the second stage by a global evaluation function and those found most promising are added to the partial solution. This procedure is repeated on a certain number of parallel paths (beamwidth). Hence, the number of solutions saved (or number of nodes expanded) at any level of the tree is equal to size of the beamwidth. The values of the filterwidth and beamwidth are usually determined empirically. In most cases, an iterative procedure is used by increasing values until the point beyond which neither the filterwidth nor the beamwidth improves the quality of the solution. In this study, the values of filterwidth and beamwidth are set to 5 and 3 based on pilot runs.

4. Reactive scheduling policies

We classify the scheduling decisions into two categories: *when-to-schedule* and *how-to-schedule*. *When-to-schedule,* determines the timing between two consecutive scheduling points. *How-to-schedule,* determines the way of generating feasible schedule. We further define three *when-to-schedule* policies and two *when-to-schedule* policies.

When-to-schedule policies considered in this paper are fixed sequencing, periodic review, continuous review. In the *fixed sequencing* approach, a schedule is generated only once at the beginning of the scheduling period and not updated later other than a simple time shifting in the Gantt chart. This policy assumes that the system recovers from the negative effects of interruptions (breakdowns, new job arrivals, etc) by itself. According to the *periodic review* policy, the system is monitored periodically and rescheduling is invoked at the beginning of time points. As discussed in Sabuncuoglu and Karabuk (1999), the periodic policy can be implemented in two ways: (1) fixed time interval and (2) variable time interval. In the *fixed time* interval method, the review periods are equally spaced points in time (i.e. at the beginning of

When-to-schedule	
PERIOD	– Periodic review with fixed time interval. – Fixed sequencing.
RATIO	– Periodic review with variable time interval.
ARRIVAL	– Continuous review.
How-to-schedule	
FULL SCHEDULE	– Scheduling all the jobs available.
PARTIAL SCHEDULE	– A certain percentage of the jobs (subset of jobs) in the system is scheduled.

Table 1. Reactive scheduling policies.

every shift, day, week, etc). According to the *variable time* interval method, the time between two scheduling points is not constant, but rather depends on the percentages of jobs processed or total processing time realized on all machines in the system. Thus, the *variable time* interval method is more responsive to the state of the system (and the current production rate) than the *fixed time* interval method. In the *continuous review*, the system is monitored continuously and rescheduling is triggered in response to a change in the system (new job arrival or machine breakdown). In the literature, this policy is also called *event-driven scheduling* policy (Ovacik and Uzsoy 1992). This policy can be implemented in a way that it responds to a certain number of arrivals or machine breakdowns rather than responding to every arrival or breakdown. These three policies are listed in table 1. PERIOD corresponds to the periodic review policy. Very large values of PERIOD correspond to the fixed sequencing policy. RATIO implements the variable time increment method in such a way that rescheduling is triggered when a determined percentage of the scheduled jobs are processed in the system. In this context, the ARRIVAL policy implements the continuous review policy.

In terms of how-to-scheduling, we use full scheduling and partial scheduling. The feature of our beam search algorithm is such that partial schedules can be easily generated once the length of the schedule is properly defined. The length can be measured either in terms of clock time or percentage of the jobs. In our study, we use the latter approach and use the percentage of the jobs to be scheduled at each decision point. A full schedule corresponds to the scheduling all the jobs available in the system (that is 100%). Similarly, a partial schedule corresponds to a schedule of a certain percentage of jobs (subset of jobs). In our simulation experiments, we compare the full scheduling with the 50% partial scheduling. In some cases, we also measure the system performance at other percentage levels.

5. System considerations and experimental conditions

An FMS studied in this paper consists of six machines each with buffer capacity, and one load/unload (L/U) station. Parts are transferred by three AGVs in the system. Interarrival time is exponential with mean 55. Each job has either five or six operations with equal probability and each operation is assigned to a different machine. Machine loads are nearly equal. Operation times are drawn from a 2-Erlang distribution with mean 55. The performance of the proposed algorithm is measured under various operating conditions with the following experimental factors: (1) buffer capacity (Q), (2) sequence flexibility (SF), (3) routing flexibility (RF), (4) tardiness factor (TF), (5) process time variation (PV), (6) machine breakdown level (e). Among

Factor	Low	High
Queue Capacity (Q)	10	100
Routing Flexibility (RF)	1	2
Sequence Flexibility (SF)	0	1

Table 2. Internal factors and their levels.

them, buffer level, sequence flexibility, routing flexibility and tardiness are considered as *internal factors*. The other factors (process time variation and machine breakdown level) are called *external factors*. For each factor, two levels (low and high) are considered in the experiments. The low and high levels for internal factors are given in table 2.

The queue capacity of the machines is set to 10 and 100, corresponding to finite and infinite values. Routing flexibility (RF) is defined as the average number of machines on which a particular operation can be processed. The value is set to 1 and 2 for low and high levels of the factor, respectively. We assume that the first assigned machine is the ideal machine with the least processing time. The processing time on the alternative machine is computed by adding a random number to the processing time of the operation on the ideal machine. Sequence flexibility (SF) is an indicator of precedence relationships between operations of the job. Specifically, operations of a job are viewed as nodes on an acyclic graph. The density of precedence arcs on this graph determines the degree of sequence flexibility. Its equation is:

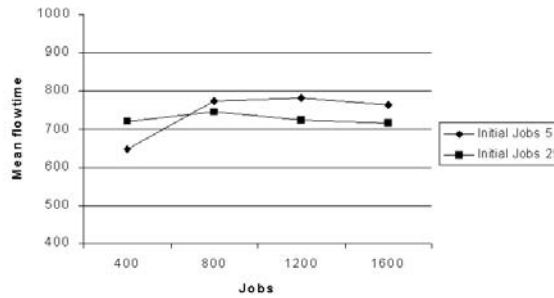
$$\text{SFM} = 1.0 - (2 \times \text{all precedence arcs})/(n \times (n-1)),$$

where n is the number of operations. The SFM value ranges between 0.0 and 1.0. The closer the SFM to 1.0, the higher the sequence flexibility a job possesses. We consider the extreme levels 0.0 and 1.0 for low and high sequence flexibilites, respectively.

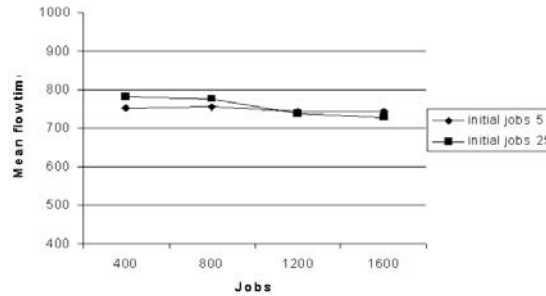
Due dates are based on total work content (TWK) rule. According to this rule, the due date of a job is determined by multiplying total work content of a job by a constant multiplier so that the desired TF value is achieved. In our study, they are assigned such that the tardiness factor (TF) is approximately fixed at 80%. The performance of the algorithm is tested for mean flowtime criterion. The local evaluation function for the mean flow-time case is LWRK (least work remaining). In addition, we use the LWRK rule for online scheduling.

6. Computational results

In this section, we present the results of simulation experiments. First we give a brief summary of the results obtained from pilot runs. In our study, we are mainly interested in seeing the effects of external factors such as dynamic job arrivals, process time variation, and machine breakdowns on scheduling policies (*when-to-schedule* policies: PERIOD, RATIO and ARRIVAL; *how-to-schedule* policies: full scheduling, and partial scheduling) and schedule generation schemes (offline and online). In order to keep the computational efforts at a reasonable level, we conduct some pilot experiments to set the values of some internal factors (i.e. queue capacity, flexibility). First, we simulate the system with two different initial job populations: 5 and 25. As can be seen in figure 2, the size of initial job population does not affect the long-term performance of the system. The system reaches steady state at nearly



(a)



(b)

Figure 2. Comparison of two different initial job population sizes for high SF and RF. (a) Beam search (with period length 200). (b) Dispatch rule.

the same times, even with the dispatching rule (i.e. using LWRK rule – least work remaining). Thus we take simulation runs with the initial job size of 5.

Second, we test different buffer capacities. In our pilot runs, we observe that the system sometimes experiences deadlock situations. Hence, we set the finite buffer capacity to 10 to avoid excessive amount of computation times. The buffer capacity of 100 is used to represent very large buffer capacity (i.e. infinite buffer capacity level). We test these two buffer sizes with scheduling policies ARRIVAL_1 (A_1) and the LWRK dispatch rule. A_1 refers to the ARRIVAL policy with its parameter 1 that corresponds to scheduling at every arrival. As seen in figure 3, the system performance is only slightly improved when the value of buffer capacity is set too large (i.e. unlimited buffer capacity). This can be seen in both the offline and online cases. The same observations are made with the PERIOD_200 (P_200) and RATIO_100 (R_100) policies. Hence, we decided to continue with the buffer size 10 in the rest of experiments. Our pilot experiments also suggest that we should use both the low and high flexibility levels in the rest of the experiments.

We also noted that the performance of the system is improved considerably as the scheduling frequency increases. For example, RATIO_25 with more frequent update than RATIO_100 yields better results. ARRIVAL_1 is better than ARRIVAL_5. Similarly, PERIOD_200 is better than PERIOD_800 (figure 4). This finding in the dynamic environment is consistent with the results of the previous studies in the static environments (Sabuncuoglu and Karabuk 1999; Sabuncuoglu and Bayiz 2000).

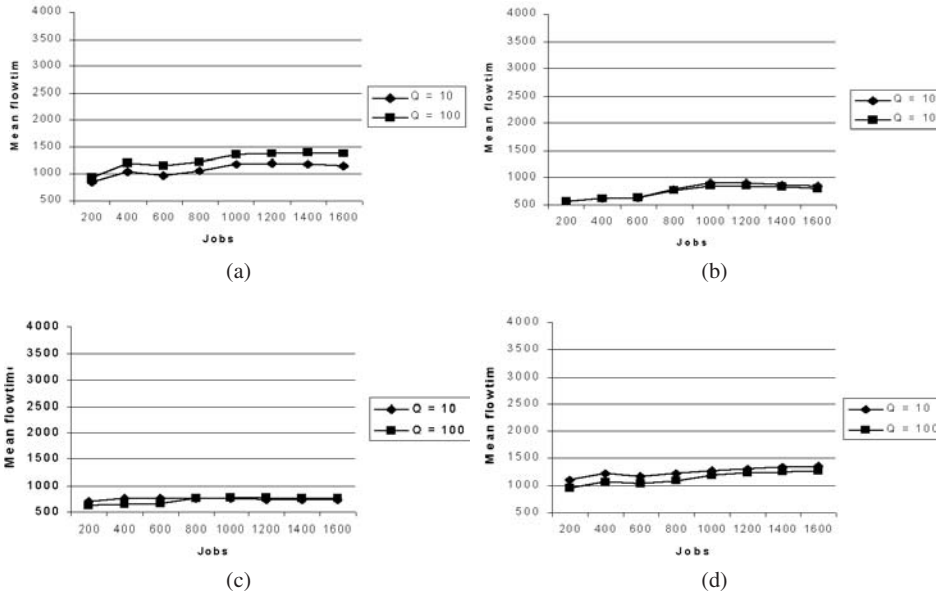


Figure 3. Comparison of queue capacities for different scheduling policies. (a) A_1 for low SF and RF; (b) A_1 for high SF and RF; (c) Dispatch for low SF and RF; (d) Dispatch for high SF and RF.

6.1. *Results in a dynamic environment*

In this section, we now compare three when-to-schedule (ARRIVAL, PERIODIC, and RATIO) and two how-to-schedule (full scheduling and partial scheduling) policies in a dynamic environment (i.e. dynamic job arrivals).

6.1.1. *How-to-schedule policies*

As stated earlier, we consider two cases: (1) full scheduling (corresponds to 100%), and (2) partial scheduling (corresponds to 50%). The results of the simulation experiments are presented in figure 5. As expected, full scheduling (100% job scheduled) yields better performance than partial scheduling (50% job rescheduled). This is observed for each of the *when-to-schedule* policies. But we note that the full scheduling requires considerably high computational times.

We further investigate the difference between partial scheduling and full scheduling at various scheduling frequencies. The parameters of the PERIODIC and RATIO policies are adjusted according to the ARRIVAL policy so that they have the same frequency levels. For that reason, A_x is displayed in the horizontal axis in figure 6. In general, the ARRIVAL policy is more affected by scheduling frequency than the RATIO and PERIODIC policies because it displays the higher envelope in these curves.

We also study the affect of various scheduling frequencies on the system performance. Sample results are given for the ARRIVAL policy in figure 7. In general, as the scheduling frequency decreases, the difference between F-HIGH and F-LOW decreases for partial scheduling. However, it is nearly constant for full scheduling. We note that for scheduling frequencies lower than A_12, both F-HIGH and F-LOW show nearly the same performances. This is because the

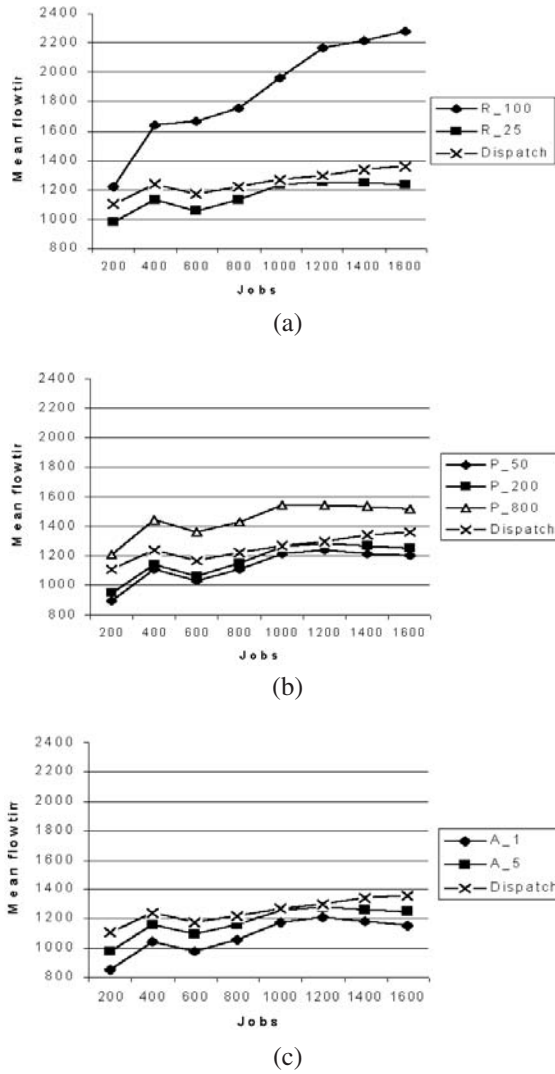


Figure 4. Comparison of different scheduling frequencies for F-LOW. (a) RATIO policy; (b) PERIODIC policy; (c) ARRIVAL policy.

search space is much smaller with the partial scheduling and the scheduling algorithm cannot use the flexibility effectively for the improved system performance.

We also compare the dispatching rule with the beam search algorithm for different frequency levels. As can be seen in figure 7, the beam search algorithm in the full scheduling mode yields better performance than the dispatching rule when the scheduling frequency is higher than A_9 (i.e., A_1, and A_6) and the flexibility is low. However, when the partial scheduling is in use, the algorithm performs better than the simple dispatch rule for the high scheduling frequencies (A_3 and A_1) and the flexibility is low. In the high flexibility case, the dispatch rule performs better than the beam search algorithm when it is implemented in the partial scheduling mode. In the full scheduling case, the offline algorithm with the ARRIVAL policy

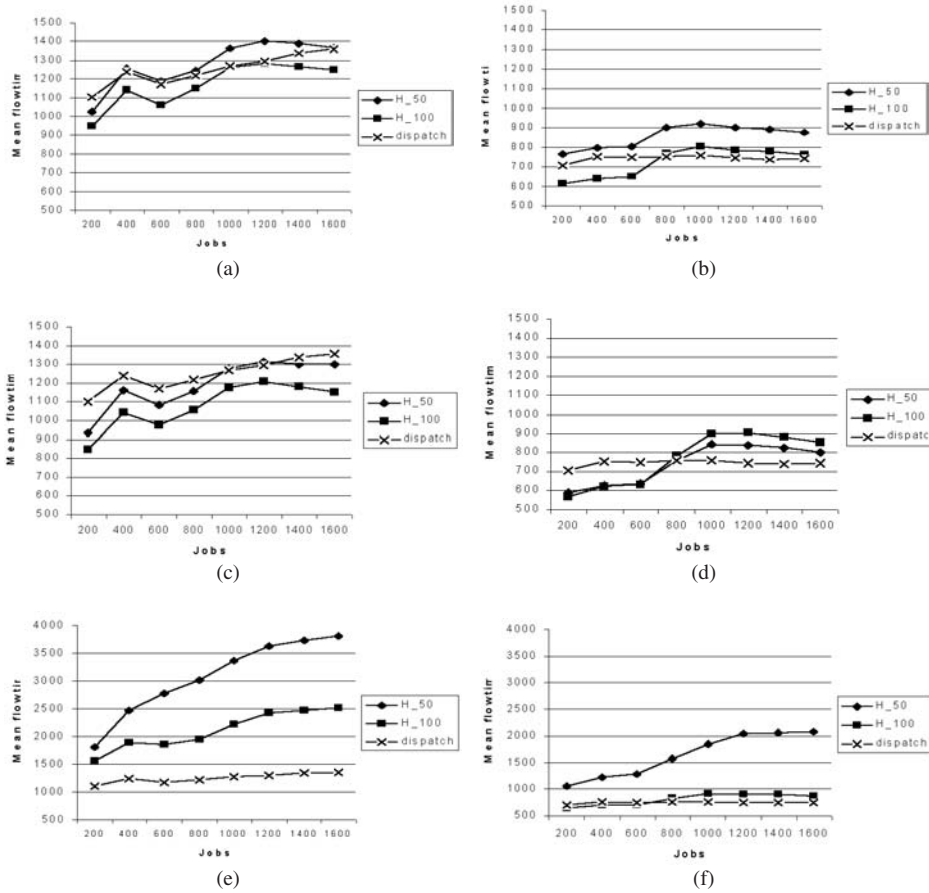


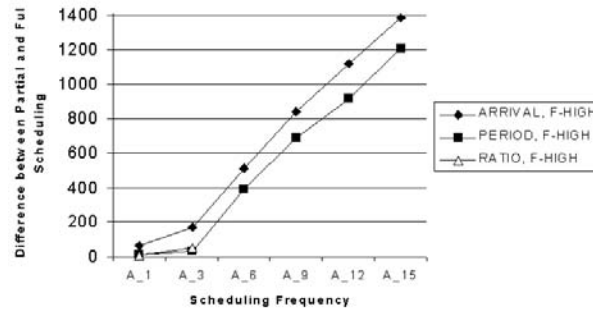
Figure 5. Comparison of how to schedule policies. (a) P_200, F-LOW; (b) P_200, F-HIGH; (c) A_1, F-LOW; (d) A_1, F-HIGH; (e) R_100, F-LOW; (f) R_100, F-HIGH.

performs better than the dispatch rule for the high frequencies (i.e., A_1 and A_3). It seems that the simple dispatch mechanism utilizes the flexibility effectively.

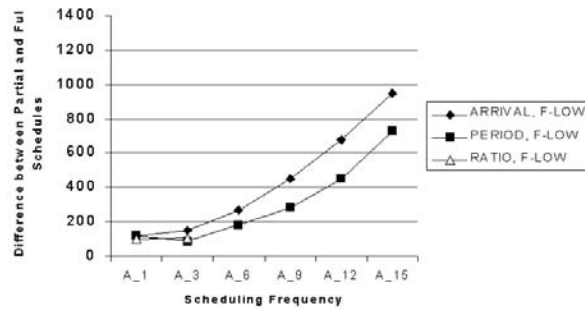
6.1.2. *When-to-schedule policies*

As stated earlier, when-to-schedule determines the timing of scheduling decisions (i.e. the time interval between two consecutive scheduling points). This time interval can be either fixed or variable. Recall that PERIOD is the fixed time interval method whereas ARRIVAL and RATIO represent variable time interval methods. These three methods are compared at various scheduling frequency levels. Again, the scheduling system is simulated for 1600 jobs. In order to compare the policies on an equal basis, we adjust their parameters in such a way that each when-to-schedule policy has approximately the same scheduling frequency (i.e. number of scheduling points are approximately equal). Specifically, we first set the parameter of ARRIVAL and then we adjust the parameters of RATIO and PERIOD accordingly. The same type of adjustment is also made for partial scheduling (H_50).

The results are summarized in figure 8. In general, the RATIO policy is better than ARRIVAL and PERIOD. As can be seen in figures 8(a) and (c), the differences



(a)

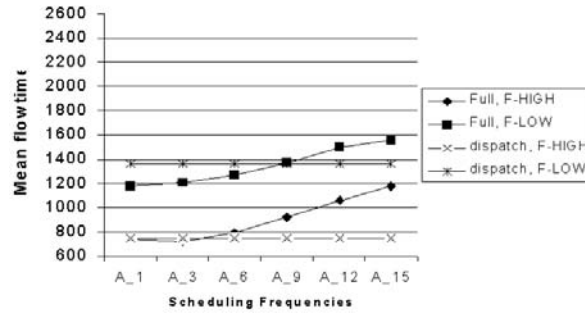


(b)

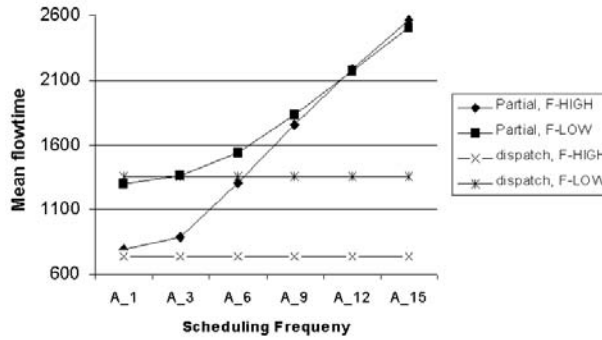
Figure 6. Differences between full and partial schedules for different scheduling frequencies. (a) F-HIGH; (b) F-LOW.

between scheduling policies are minimum when the flexibility is low (i.e. F-LOW) and the frequency of scheduling is very high (i.e. A_1 case). This is because some of the policies do not have enough opportunities to improve the system performance when the flexibility is LOW and there are too many schedule revisions. In particular, when the scheduling decisions are frequently made (i.e. A_1), the scheduling policies respond to nearly every arrival or departure and, thus, they display similar performances. This observation is valid for both full and partial scheduling. The relatively better performance of the RATIO policy can be attributed to the fact that it is more sensitive to the production rate (output process of the system).

The performance of PERIODIC and ARRIVAL are quite mixed. In general, ARRIVAL is better when it is used with the full scheduling scheme whereas PERIODIC is better than ARRIVAL with the partial scheduling. To understand this mixed behaviour, we further conduct the simulation experiments at various partial scheduling levels. The results indicate that PERIODIC is better than ARRIVAL when the partial scheduling level is low (figure 9 displays a sample of results for F-High). However, as the partial scheduling level increases, ARRIVAL gets better than PERIODIC. We also note that ARRIVAL is more sensitive to the partial scheduling levels as compared to PERIODIC. Note that the crossover point moves to the left when the scheduling frequency is high, because the difference between PERIODIC and ARRIVAL decreases as the frequency increases. Moreover, the scheduling interval is variable in the ARRIVAL policy (a long



(a)



(b)

Figure 7. Comparison of flexibilities for full and partial schedules. (a) ARRIVAL; (b) ARRIVAL.

scheduling interval is followed by a shorter interval or a longer interval) as compared to the fixed scheduling interval in the PERIODIC policy.

When the scheduling interval is too long, the system can process all the jobs scheduled by the partial schedule and waits idle. As compared with the fixed interval of the PERIODIC policy, if the scheduling interval is too long in the ARRIVAL policy, unnecessary idle times are inserted into the schedule since the machines can process all the jobs scheduled by the partial scheduling for the ARRIVAL policy. For that reason, ARRIVAL displays inferior performance when the partial scheduling level is low. In short, we can conclude that the RATIO policy, which relies on the output process, is better than the ARRIVAL policy (which relies on the input process) and the PERIODIC policy (which relies neither on input nor output process). We also test if these differences are significant. The results indicate that the offline scheduling scheme is significantly better than the online dispatching rule in the high flexibility case. In the low flexibility case, no significant difference is identified. Similar observations are made for the RATIO and PERIODIC policies.

6.2.1. *Processing time variation*

In a typical manufacturing environment, actual processing times realized on the shop floor can be substantially different than the ones used in the preparation of schedules. These variations, called processing time variations (PVs), degrade the

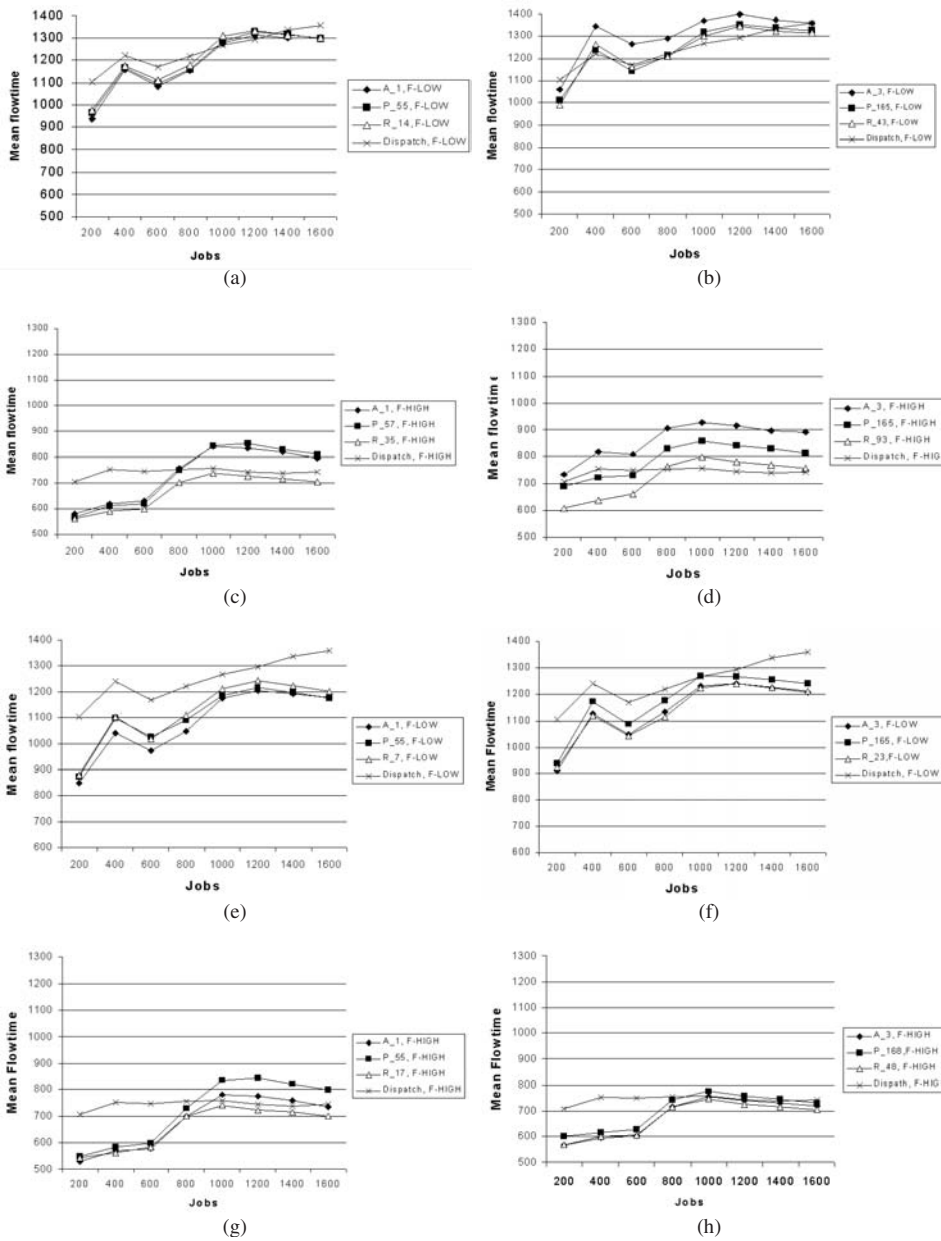


Figure 8. Comparison of when to schedule policies for partial and full schedules. (a) Partial schedule at low flexibility; (b) partial schedule at low flexibility; (c) partial schedule at high flexibility; (d) partial schedule at high flexibility; (e) full schedule at low flexibility; (f) full schedule at low flexibility; (g) full schedule at high flexibility; (h) full schedule at high flexibility.

quality of the scheduling decision as well as the system performance. In this section, we will consider this important issue. In the simulation experiments, we generate the estimated processing times from a 2-Erlang distribution. Actual processing times are obtained from these estimates by perturbing them by a certain amount when the

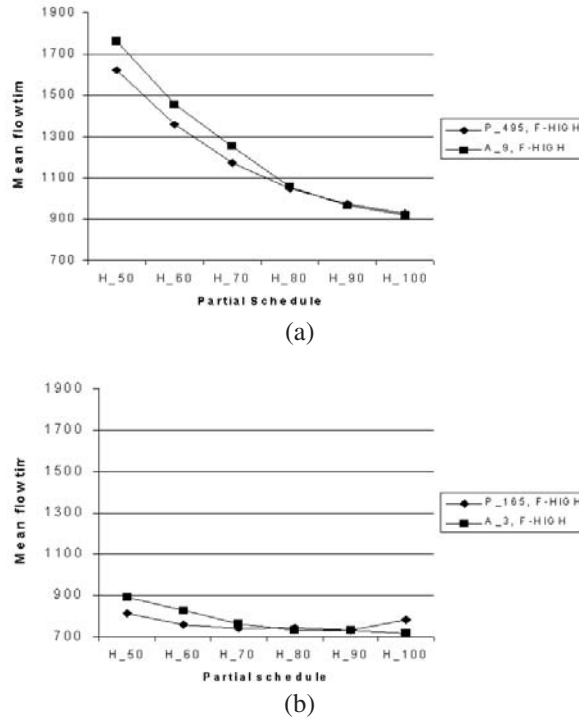


Figure 9. Comparison of PERIODIC and ARRIVAL schedules for different partial schedules. (a) Late response; (b) fast response.

schedule is executed in the simulation module. In fact, actual times are generated from a truncated normal distribution with mean equal to the estimated processing time and coefficient of variation (CV) of 0.4. Again, we take the simulation runs for the three when-to-schedule (ARRIVAL, PERIODIC and RATIO) policies and two how-to-schedule policies (partial scheduling and full scheduling). Figure 10 presents the results for the scheduling frequencies corresponding to A_3. The following observations are made.

First, in the no-PV case, the performance of the reactive policies in the offline full scheduling mode are better than a simple dispatch rule for the scheduling frequency A_3 (figures 10(a) and (c)). What is surprising is that the simple dispatch rule performs better than the offline algorithm for the A_9 case (see figures 10(e) and (g) for even the low scheduling frequency case). This suggests that the simple rules, which are commonly used in practice, are quite effective in dynamic and stochastic environments.

Second, the performances of scheduling methods and the dispatch rule deteriorates as PV increases. The interesting point is that the offline algorithm is more sensitive to process time variation than the simple dispatch rule. As seen in figures 10(b), (d), (f) and (h), the performance of the offline algorithm deteriorates more than the online dispatch rule. Specifically, for both the partial and full scheduling, the simple LWRK rule performs better than the offline algorithm with three reactive policies at the scheduling frequencies A_3 and A_9. Note that the difference even becomes larger when the scheduling frequency decreases from A_3 to A_9.

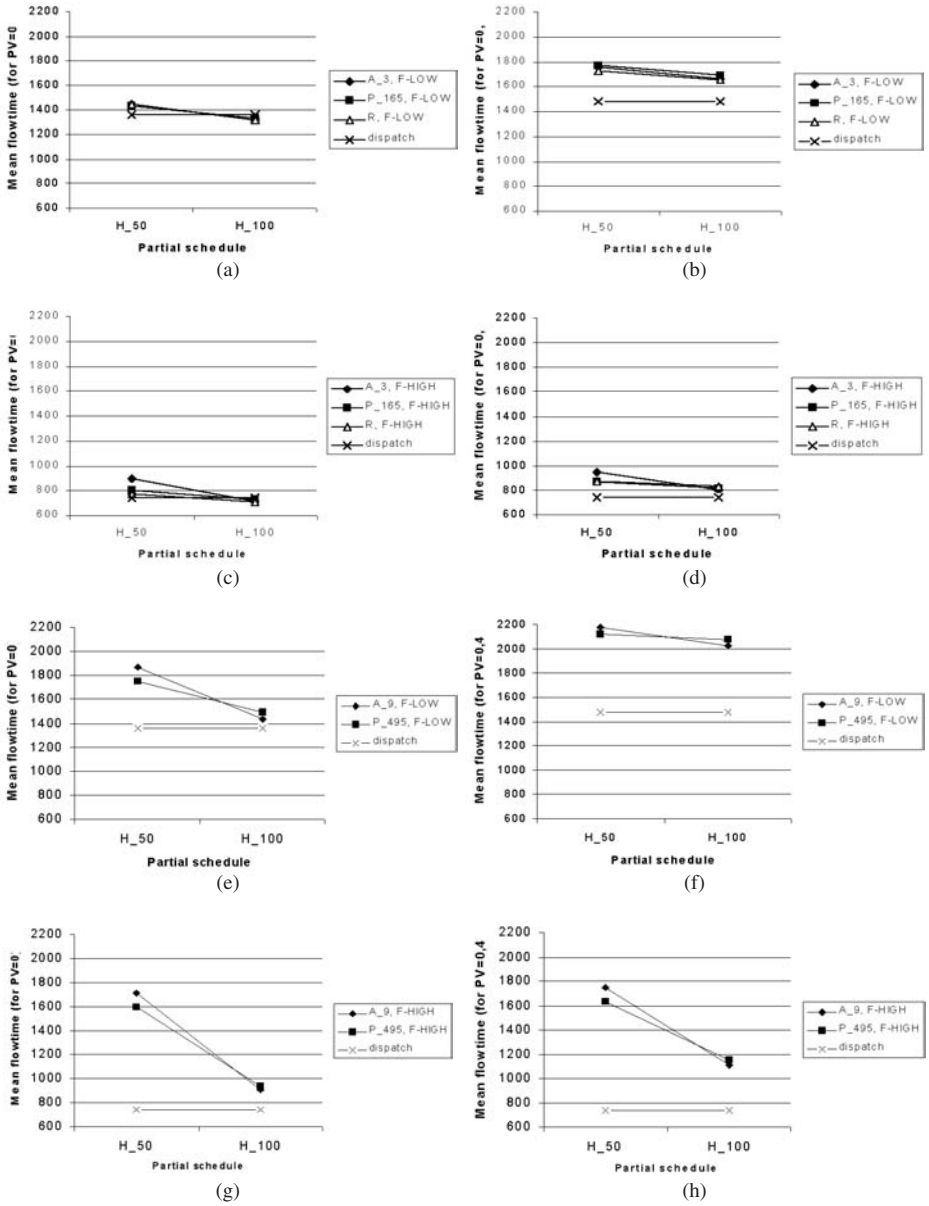


Figure 10. Mean flowtimes for the ARRIVAL, PERIODIC, RATIO policies for PV = 0 and PV = 0.4. (a) PV = 0; (b) PV = 0.4; (c) PV = 0; (d) PV = 0.4; (e) PV = 0; (f) PV = 0.4; (g) PV = 0; (h) PV = 0.4.

Third, similar to the previous findings in the static environments, the system performance is better with the full scheduling scheme than the partial scheduling scheme (see figures 10(e) and (g)).

Fourth, the PERIODIC policy performs better than the ARRIVAL policy with partial scheduling. However, the ARRIVAL policy yields better results in the full schedule case. Overall, the RATIO policy seems to be the best.

Fifth, the difference between partial and full scheduling decreases with PV because full scheduling is more affected from PV as compared to partial scheduling.

6.2.2. *Machine breakdowns*

We also test the scheduling policies under machine breakdowns. This is modelled by the busy time approach (Law and Kelton 2000). The parameters are adjusted in such a way that the system overall availability level is 90%, which gives the long run ratio of a machine busy time to busy plus down time.

A summary of the results is given in table 3. First, as intuitively expected, breakdowns have negative impacts on the performances of scheduling policies. Second, the adverse effect of breakdowns is greater with the full scheduling than the partial scheduling. This is because more operations are affected in the offline mode. Third, the difference between the scheduling policies decreases with machine breakdowns (compared with the no machine breakdown case) especially when a partial schedule is implemented. Fourth, machine breakdowns have more adverse impact on the system performance than processing time variations. This is because the machine breakdown has an immediate effect on the system whereas the effect of PV on the system performance is accumulated over time. This finding is consistent with the results reported in the static environment (Sabuncuoglu and Karabuk 1999). Fifth, in the no-breakdown case the performances of the scheduling policies are significantly better with full scheduling than partial scheduling. We observe the opposite behaviour of the scheduling policies with breakdowns. That is, the system performance is improved with the partial scheduling. This counter intuitive result can be explained by the fact that the benefit of using full scheduling diminishes when there are machine breakdowns (i.e. the entire schedule may become totally useless with machine breakdowns).

	No Br.	Br.		No Br.	Br.
Low scheduling frequency case					
F-LOW	P_495	P_495	F-HIGH	P_495	P_495
H_50	1749	3628	H_50	1594	2279
H_100	1491	4237	H_100	939	2381
F-LOW	A_9	A_9	F-HIGH	A_9	A_9
H_50	1868	3969	H_50	1710	2861
H_100	1430	4162	H_100	908	2366
High scheduling frequency case					
F-LOW	P_165	P_165	F-HIGH	P_165	P_165
H_50	1424	3262	H_50	800	2022
H_100	1330	3452	H_100	733	2048
F-LOW	A_3	A_3	F-HIGH	A_3	A_3
H_50	1442	3097	H_50	897	2075
H_100	1314	3345	H_100	717	1978
F-LOW	R	R	F-HIGH	R	R
H_50	1444	3182	H_50	773	1889
H_100	1316	3208	H_100	706	1962

Table 3. Results under machine breakdowns.

Sixth, the RATIO policy seems to be the best among all three policies. Seventh, the offline algorithm is more sensitive to breakdowns than the online dispatching rule. This means that, under certain conditions, where the system experiences more frequent breakdowns and higher PV, the differences between two approaches decrease. This observation also confirms the intuition in practice that the potential benefit of the optimum seeking algorithms in real manufacturing environments is not as good as what is planned due to unexpected interruptions and the dynamic nature of systems.

7. Concluding remarks

In this paper we address the main issues of dynamic and stochastic scheduling problems. In the first part, we briefly review the existing studies in the literature. In the second part, we propose a simulation-based scheduling system for a dynamic and stochastic FMS environment. In the third part, we analyze the major scheduling issues and measure the performances of different reactive scheduling policies, compare full versus partial scheduling schemes, and online versus offline scheduling methods. The following conclusions can be drawn from our study.

First, as the frequency of rescheduling increases, the performance of the system gets better. In a way, the system performance is proportional to the reschedule frequency. However, we also noted that the positive contribution of frequent scheduling becomes marginal after some point. Second, the timing of the response is also as important as the scheduling frequency. Our results indicate that the variable time interval method is better than the fixed time interval method. Third, full rescheduling is generally better than partial rescheduling at a cost of higher CPU times. This conclusion is also consistent with the results drawn by Sabuncuoglu and Bayiz (2000). Partial scheduling becomes very competitive when there are severe machine breakdowns and the flexibility is low. Fourth, the online scheduling mechanism represented by the dispatch rule is more robust to process time variation and machine breakdowns than the offline scheduling algorithm. Moreover, the dispatching rule requires less CPU time compared with the offline algorithm. Hence it would be more beneficial to use the online scheduling systems in dynamic and stochastic environments.

The results presented in the paper should be interpreted with reference to the assumptions and experimental conditions described earlier. Even though some of our results are valid for job shop systems due to the fact that FMS is a more general version of job shop systems, there is a need for further research to test the policies under different conditions. One such condition is to test different efficiency levels for the machine breakdowns. Moreover, the effects of different durations of mean machine up and down times for the same efficiency level can be analyzed. New rescheduling methods (i.e. robust scheduling) can be developed and tested in stochastic and dynamic environments. The beam search algorithm can be tested with different local and global evaluation functions. Finally, the simulation study can be extended to cover other combinations of experimental factors (i.e. different machine load levels, AGV speeds, tardiness factor).

References

Bean, J. C., Birge, J., Mintenhal, J. and Noon, C., 1991, Matchup scheduling with multiple resources, release dates, and disruptions. *Operations Research*, **39**, 470–483.

Church, L. and Uzsoy, R., 1992, Analysis of periodic and event driven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing*, **5**, 153–163.

Daniels, R. and Kouvelis, P., 1995, Robust scheduling to hedge against processing time uncertainty in single stage production. *Management Science*, **41**, 363–376.

Dutta, A., 1994, Reacting to scheduling exceptions in FMS environments. *IIE Transactions*, **22**, 300–314.

Fox, M. S. and Smith, S. F., 1984, ISIS-knowledge based system for factory scheduling. *Expert Systems*, **1**, 25–49.

He, Y., Smith, M. and Dudek, R., 1994, Effects of inaccuracy of processing time estimation on effectiveness of dispatch rules. *Third Industrial Engineering Research Conference*, pp.308–313.

Jeong, K. J. and Kim, Y. D., 1998, A real-time scheduling mechanism for a flexible manufacturing system: using simulation and dispatching rules. *International Journal of Production Research*, **36**(9), 2609–2626.

Kim, M. and Kim, Y., 1994, Simulation based real-time scheduling in a flexible manufacturing system. *Journal of Manufacturing System*, **13**, 85–93.

Kutanoglu, E. and Sabuncuoglu, I., 2001a, Experimental investigation of iterative simulation-based scheduling in dynamic and stochastic job shop. *Journal of Manufacturing Systems*, **20**(4), 264–278.

Kutanoglu, E. and Sabuncuoglu, I., 2001b, Routing-based reactive scheduling policies for machine failures in job shops. *International Journal of Production Research*, **39**(14), 3141–3158.

Law, A. M. and Kelton, W. D., 2000, *Simulation Modelling and Analysis* (McGraw-Hill).

Lawrence, S. R. and Sewell, E. C., 1997, Heuristic, optimal, static, and dynamic schedules when processing times are uncertain. *Journal of Operations Management*, **15**, 71–82.

Lejmi, T. and Sabuncuoglu, I., 2002, Effect of load and processing time variation on the effectiveness of scheduling rules. *International Journal of Production Research*, **40**(4), 945–974.

Leon, V. J., Wu, S. D. and Storer, R. H., 1994, Robustness measures and robust scheduling for job shops. *IIE Transactions*, **26**, 32–43.

Mehta, S. and Uzsoy, R., 1998, Predictable scheduling of a job shop subject to breakdowns. *IEEE Transactions on Robotics and Automation*, **14**, 365–378.

Mehta, S. and Uzsoy, R., 1999, Predictable scheduling of a single machine subject to breakdowns. *International Journal Computer Integrated Manufacturing*, **12**, 15–38.

Muhlemann, A. P., Lockett, A. G. and Farn, C. K., 1982, Job shop scheduling heuristics and frequency of scheduling. *International Journal of Production Research*, **20**, 227–241.

Nelson, R. T. and Holloway, C. A., 1977, Centralized scheduling and priority implementation heuristics for a dynamic job shop model with due dates and variable processing times. *AIIE Transactions*, **l9**, 95–102.

Ovacik, I. M. and Uzsoy, R., 1992, Analysis of periodic and event-driven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing*, **5**, 153–163.

Ovacik, I. M. and Uzsoy, R., 1994, Rolling horizon algorithms for a single machine dynamic scheduling problem with sequence-dependent set-up times. *International Journal of Production Research*, **32**, 1243–1263.

Sabuncuoglu, I., 1998, Scheduling with neural networks – a review of the literature and new research directions. *Production and Inventory Control*, **9**(1), 2–12.

Sabuncuoglu, I. and Bayiz, M., 2000, Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research*, **126**, 567–586.

Sabuncuoglu, I. and Karabuk, S., 1998, A beam search algorithm and evaluation of scheduling approach for flexible manufacturing systems. *IIE Transactions*, **30**, 179–191.

Sabuncuoglu, I. and Karabuk, S., 1999, Rescheduling frequency in an FMS with uncertain processing times and unreliable machines. *Journal of Manufacturing Systems*, **18**(4), 268–283.

Shaw, M., Park, S. and Raman, N., 1992, Intelligent scheduling with machine learning capabilities: the induction of scheduling knowledge. *IIE Transactions*, **24**, 156–168.

SMITH, S. F., OW, P., POTWIN, J., MUSCETTOLA, N. and MATTHYS, D. C., 1990, An integrated framework for generating and revising factory schedules. *Journal of Operational Research Society*, **41**, 539–552.

SVESTKA, A. J. and ABUMAZIAR, R. J., 1997, Rescheduling job shops under random disruptions. *International Journal of Production Research*, **35**, 2065–2082.

SZELKE, E. and KERR, R., 1994, Knowledge-based reactive scheduling. *Production Planning and Control*, **5**, 124–145.

WU, L. and STORER, R., 1994, Robustness scheduling for job shop. *IIE Transactions*, **26**, 32–41.

WU, D., STORER, R. H. and CHANG, P., 1993, One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers Operations Research*, **20**, 1–13.

YAMAMOTO, M. and NOF, S. Y., 1985, Scheduling/rescheduling in the manufacturing operating system environment. *International Journal of Production Research*, **23**, 705–722.